Second Midterm Exam (a) MCS 360 Jan Verschelde Wednesday 22 November 2017

NAME : ***answers***

The exam is closed book, no notes and no computer.
All your answers to the questions below must be submitted on paper.
Write your name on this sheet and submit it with your answers.
Please do not ask questions during the exam.

1. Schröder numbers can be defined as

$$R_0 = 1 \quad \text{and for } n \geq 1: \quad R_n = R_{n-1} + \sum_{k=0}^{n-1} R_k R_{n-1-k}.$$

   (a) Explain why the application of the definition in a recursive function to compute R_n will be wasteful. In your answer, write complete sentences.

   (b) Write an efficient recursive function `R` to compute R_n using memoization.

answer:

(a) The definition is inefficient because to compute R_n, we make $2n$ recursive calls and many of these calls lead to redundant, repeated computations. This generates a tree of calls with $O((2n)^n)$ leaves. All leaves have value one (the base case in the recursion) and many nodes appear an exponential number of times with the same value.

(b)
```
size_t R ( size_t n )
{
   static vector<size_t> mem;

   while(mem.size() <= n) mem.push_back(-1);

   if(mem[n] != -1)
      return mem[n];
   else
   {
      size_t result;

      if(n==0)
         result = 1;
      else
      {
         result = R(n-1);
         for(size_t k=0; k<n; k++)
            result = result + R(k)*R(n-1-k);
      }
      mem[n] = result;
      return result;
   }
}
```

2. Consider a vector of strings, which holds the names of fruit items.
   For example: `"apple"`, `"banana"`, `"lemon"`, `"grape"`, `"orange"`.

   Write a C++ function to show all combinations of k items, for k between 1 and the size of the vector. The function writes all combinations to screen, one line per combination. For the example above, if k equals 3, then the output starts with

```
apple banana lemon
apple banana grape
apple banana orange
apple lemon grape
etc. ...
```

   Observe that every fruit item occurs only once. Your function should work for any size of the vector of strings and for any k between 1 and the size of the vector.

   (a) Write the prototype of your function and document all its parameters.

   (b) Give the definition of your function.

answer:

(a) The prototype of the function is

```
void choose
 ( size_t k, size_t cnt, size_t start, vector<string> things, string accu );
/*
   Shows all choices of k things, using accu as accumulating parameter.
   The five parameters have the following meaning:
   k      : number of things to choose;
   cnt    : counts the number of things already chosen;
   start  : start index in things to begin the choice;
   things : strings to choose from;
   accu   : accumulates the string of choices. */
```

(b) The definition of the function is

```
void choose
 ( size_t k, size_t cnt, size_t start, vector<string> things, string accu )
{
   if(cnt == k)
      cout << accu << endl;
   else
      for(size_t i=start; i<things.size(); i++)
         choose(k, cnt+1, i+1, things, accu + " " + things[i]);
}
```

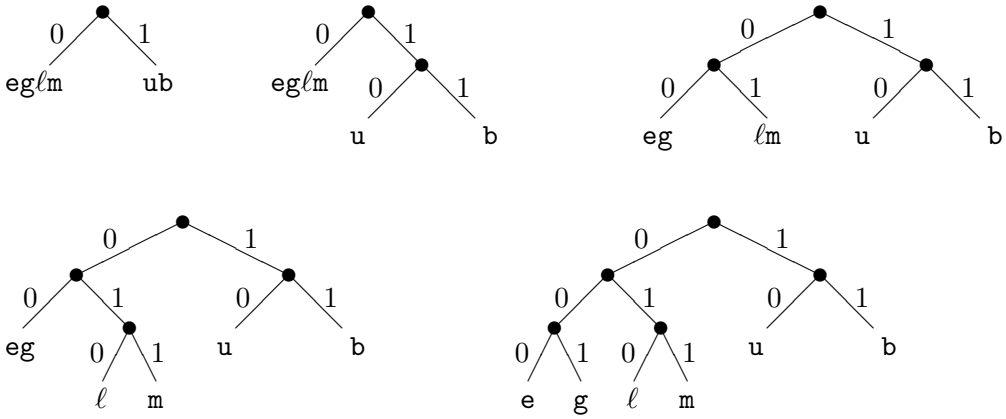
3. Create a Huffman code for the word `"bubblegum"`.
   Show all stages in your solution.

   answer:

   (a) Contracting frequency tables:

·	$f(\cdot)$	·	$f(\cdot)$	·	$f(\cdot)$	·	$f(\cdot)$	·	$f(\cdot)$
e	1	ℓ	1	eg	2	u	2	egℓm	4
g	1	m	1	ℓm	2	b	3	ub	5
ℓ	1	eg	2	u	2	egℓm	4		
m	1	u	2	b	3				
u	2	b	3						
b	3								

   (b) Making the tree:



   (c) Our Huffman code for `"bubblegum"` is

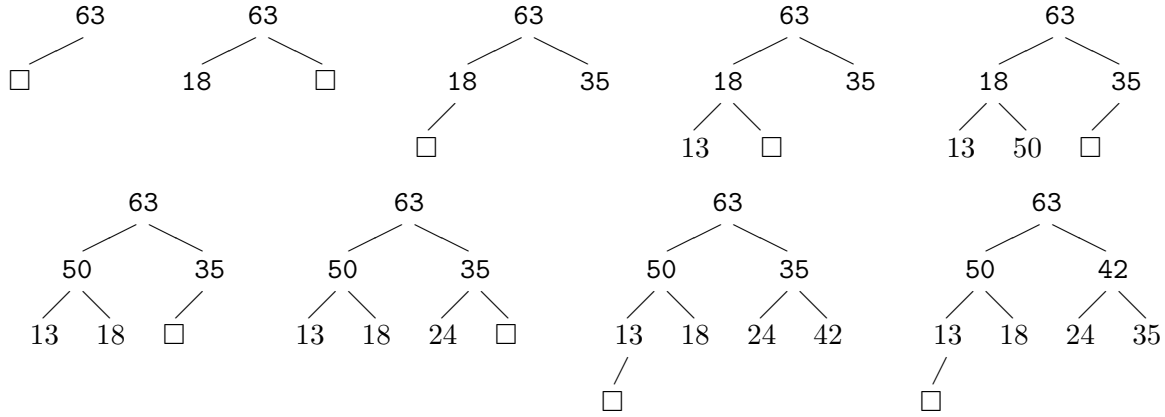
b	u	ℓ	e	g	m
11	10	010	000	001	011

   and `"bubblegum"` is encoded as `1110111101000000110011`.

4. Consider the numbers 63, 18, 35, 13, 50, 24, 42. Sort the numbers in decreasing order (largest number first, smallest number last) using a heap (or priority queue).
Show ***all*** stages in the evolution of the heap.

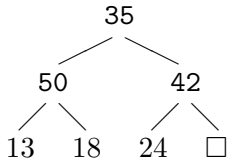
answer:

(a) Pushing the numbers to the heap:

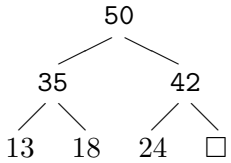


(b) Popping the top from the heap (S = sorted sequence):

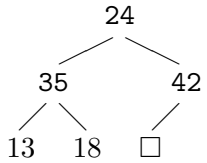
S: 63



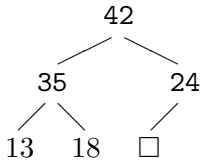
S: 63



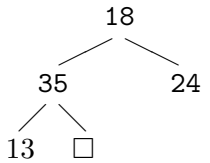
S: 63, 50



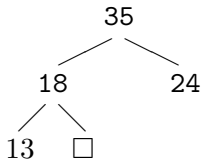
S: 63, 50



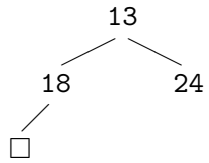
S: 63, 50, 42



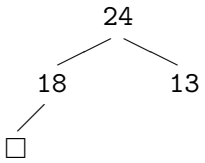
S: 63, 50, 42



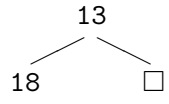
S: 63, 50, 42, 35



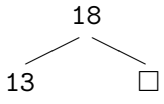
S: 63, 50, 42, 35



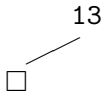
S: 63, 50, 42, 35, 24



S: 63, 50, 42, 35, 24

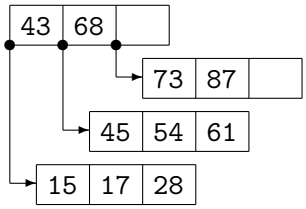


S: 63, 50, 42, 35, 24, 18



S: 63, 50, 42, 35, 24, 18, 13

5. Consider the 4-tree:

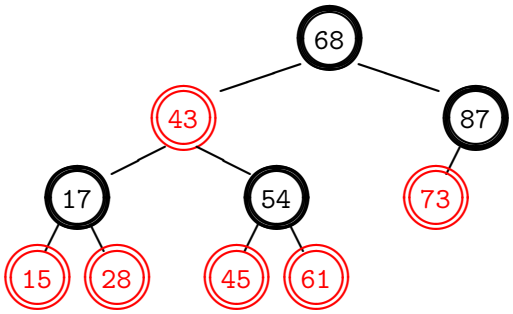


(a) Draw the equivalent red-black tree to the 4-tree above.

(b) Draw the 4-tree after inserting 56 in the 4-tree above.

answer:

(a) The equivalent red-black tree:



(b) Inserting 56: